
* ITWOM version 2.0t, September 26, 2010 File: itwom2.0t.cpp *
* Provenance: Further test version of itwom2.0m re adj to Hrzn range factors *
* 1. This file is based on a thorough debugging, completion, and update of the *
* ITM, based on an original, public domain version of this file obtained from: *
* ftp://flattop.its.bldrdoc.gov/itm/ITMDLL.cpp prior to May, 2007. C++ routines *
* for this program are taken from a translation of the FORTRAN code written by *
* U.S. Department of Commerce NTIA/ITS Institute for Telecommunication Sciences *
* Irregular Terrain Model (ITM) (Longley-Rice). *
* 2. The Linux version of this file incorporates improvements suggested by a *
* study of changes made to file itm.cpp by J. D. McDonald to remove Microsoft *
* Windows dll-isms and to debug an ambiguity in overloaded calls. *
* 3. The Linux version of this file also incorporates improvements suggested by *
* a study of further modifications made to itm.cpp by John A. Magliacane to *
* remove unused variables, unneeded #includes, and to replace pow() statements *
* with explicit multiplications to improve execution speed and accuracy. *
* 4. On August 19, 2007 this file was modified by Sid Shumate to include *
* changes and updates included in version 7.0 of ITMDLL.cpp, which was released *
* by the NTIA/ITS on June 26, 2007. With correction set SS1 and SS2: itm71.cpp. *
* 5. On Feb. 5, 2008 this file became v.1.0 of the ITWOM with the addition, by *
* Sid Shumate, of multiple corrections, the replacement of subroutines lrprop *
* and alos with lrprop2 and alos2, and the addition of subroutine saalos to *
* incorporate Radiative Transfer Engine (RTE) computations in the line of sight *
* range. *
* Update 8 Jun 2010 to modify alos to match 2010 series of IEEE-BTS *
* newsletter articles *
* Update June 12, 2010 to z version to change test outputs *
* Offshoot start date June 23, 2010 to start itwom2.0 dual version for FCC. *
* Update to 2.0b July 25 to correct if statement errors in adiff2 re two peak *
* calculations starting at line 525 *
* Development to 2.0c 8 Aug 2010 after modifying saalos and adiff for full *
* addition of saalos treatment to post obstruction calculations and debugging. *
* Modified to make 1st obs loss=5.8 only, no clutter loss considered *
* *
* This file is copyright(c) 2010 by Sid Shumate and Givens & Bell, Inc. *
* All rights reserved. Commercial use, and resale, including when compiled with *
* wrap-around software, is prohibited except under Givens & Bell, Inc. license. *
* *
*****/

```
#include <math.h>
#include <complex>
#include <assert.h>
#include <string.h>

#define THIRD (1.0/3.0)
```

```
using namespace std;
```

```
struct tcomplex
{ double tcreal;
  double tcimag;
};
```

```
struct prop_type
{ double aref;
  double dist;
  double hg[2];
  double rch[2];
  double wn;
  double dh;
  double dhd;
  double ens;
  double encc;
  double cch;
  double cd;
  double gme;
  double zgndreal;
  double zgndimag;
  double he[2];
  double dl[2];
  double the[2];
  double tiw;
  double ght;
  double ghr;
  double rph;
  double hht;
  double hhr;
```

```
double tgh;
double tsgh;
double ther;
double thenr;
int rpl;
int kwx;
int mdp;
int ptx;
int los;
};
```

```
struct propv_type
{ double sgc;
  int lvar;
  int mdvar;
  int klim;
};
```

```
struct propa_type
{ double dlsa;
  double dx;
  double ael;
  double ak1;
  double ak2;
  double aed;
  double emd;
  double aes;
  double ems;
  double dls[2];
  double dla;
  double tha;
  double test0;
  double test1;
  double test2;
  double test3;
  double test4;
};

};
```

```
int mymin(const int &i, const int &j)
```

```
{
```

```
if (i<j)
```

```
return i;
```

```
else
```

```
return j;
```

```
}
```

```
int mymax(const int &i, const int &j)
```

```
{
```

```
if (i>j)
```

```
return i;
```

```
else
```

```
return j;
```

```
}
```

```
double mymin(const double &a, const double &b)
```

```
{
```

```
if (a<b)
```

```
return a;
```

```
else
```

```
return b;
```

```
}
```

```
double mymax(const double &a, const double &b)
```

```
{
```

```
if (a>b)
```

```
return a;
```

```
else
```

```
return b;
```

```
}
```

```
double FORTRAN_DIM(const double &x, const double &y)
```

```
{
```

```
/* This performs the FORTRAN DIM function. Result is x-y
```

```
if x is greater than y; otherwise result is 0.0 */
```

```
if (x>y)
```

```
return x-y;
```

```

else
    return 0.0;
}

double aknfe(const double &v2)
{
    double a;

    if (v2<5.76)
        a=6.02+9.11*sqrt(v2)-1.27*v2;
    else
        a=12.953+10*log10(v2);
    return a;
}

double fht(const double& x, const double& pk)
{
    double w, fhtv;

    if (x<200.0)
    {
        w=-log(pk);

        if (pk<1.0e-5 || x*w*w*w > 5495.0)
        {
            fhtv=-117.0;

            if (x>1.0)
                fhtv=40.0*log10(x)+fhtv;
            }
            else
                fhtv=2.5e-5*x*x/pk-8.686*w-15.0;
        }

        else
        {
            fhtv=0.05751*x-10.0*log10(x);

            if (x<2000.0)

```

```
{  
w=0.0134*x*exp(-0.005*x);  
fhtv=(1.0-w)*fhtv+w*(40.0*log10(x)-117.0);  
}  
}  
return fhtv;  
}
```

```
double h0f(double r, double et)  
{  
double a[5]={25.0, 80.0, 177.0, 395.0, 705.0};  
double b[5]={24.0, 45.0, 68.0, 80.0, 105.0};  
double q, x;  
double h0fv, temp;  
int it;
```

```
it=(int)et;
```

```
if (it<=0)  
{  
it=1;  
q=0.0;  
}
```

```
else if (it>=5)  
{  
it=5;  
q=0.0;  
}
```

```
else  
q=et-it;
```

```
/* x=pow(1.0/r,2.0); */
```

```
temp=1.0/r;  
x=temp*temp;
```

```
h0fv=4.343*log((a[it-1]*x+b[it-1])*x+1.0);
```

```

if (q!=0.0)
h0fv=(1.0-q)*h0fv+q*4.343*log((a[it]*x+b[it])*x+1.0);

return h0fv;
}

double ahd(double td)
{
int i;
double a[3]={ 133.4, 104.6, 71.8};
double b[3]={0.332e-3, 0.212e-3, 0.157e-3};
double c[3]={ -4.343, -1.086, 2.171};

if (td<=10e3)
i=0;

else if (td<=70e3)
i=1;

else
i=2;

return a[i]+b[i]*td+c[i]*log(td);
}

double abq_los(complex<double> r)
{
return r.real()*r.real()+r.imag()*r.imag();
}

double saalos(double d, prop_type &prop, propa_type &propa)
{
double ensa, encca, q, dp, dx, tde, hc, ucrpc, ctip, tip, tic, stic, ctic, sta;
double ttc, cttc, crpc, ssnps, d1a, rsp, tsp, arte, zi, pd, pdk, hone, tvsr;
double saalosv=0.0;

q=0.0;

```

```

if (d==0.0)
{
tsp=1.0;
rsp=0.0;
d1a=50.0;
saalosv=0.0;
}
else if(prop.hg[1] > prop.cch)
{
saalosv=0.0;
}
else
{
pd=d;
pdk=pd/1000.0;
tsp=1.0;
rsp=0.0;
d1a=pd;
/* at first, hone is transmitter antenna height
relative to receive site ground level. */
hone=prop.tgh+prop.tsgh-(prop.rch[1]-prop.hg[1]);

if(prop.tgh>prop.cch) /* for TX ant above all clutter height*/
{
ensa=1+prop.ens*0.000001;
encca=1+prop.encc*0.000001;
dp=pd;

for (int j=0; j<5; ++j)
{
tde=dp/6378137.0;
hc=(prop.cch+6378137.0)*(1-cos(tde));
dx=(prop.cch+6378137.0)*sin(tde);
ucrpc=sqrt((hone-prop.cch+hc)*(hone-prop.cch+hc)+(dx*dx));
ctip=(hone-prop.cch+hc)/ucrpc;
tip=acos(ctip);
tic=tip+tde;
tic=mymax(0.0,tic);
stic=sin(tic);
}
}

```

```

sta=(ensa/encca)*stic;
ttc=asin(sta);
cttc=sqrt(1-(sin(ttc))*(sin(ttc)));
crpc=(prop.cch-prop.hg[1])/cttc;
if(crpc>=dp)
{
    crpc=dp-1/dp;
}

ssnps=(3.1415926535897/2)-tic;
d1a=(crpc*sin(ttc))/(1-1/6378137.0);
dp=pd-d1a;

}

ctic=cos(tic);

/* if the ucrpc path touches the canopy before reaching the
   end of the ucrpc, the entry point moves toward the
   transmitter, extending the crpc and d1a. Estimating the d1a: */

if(ssnps<=0.0)
{
    d1a=mymin(0.1*pd,600.0);
    crpc=d1a;
    /* hone must be redefined as being barely above
       the canopy height with respect to the receiver
       canopy height, which despite the earth curvature
       is at or above the transmitter antenna height. */
    hone=prop.cch+1;
    rsp=.997;
    tsp=1-rsp;
}
else
{
    if (prop.ptx>=1) /* polarity ptx is vertical or circular */
    {
        q=((ensa*cttc-encca*ctic)/(ensa*cttc+encca*ctic));

```

```

rsp=q*q;
tsp=1-rsp;

if (prop.ptx==2) /* polarity is circular - new */
{
    q=((ensa*ctic-encca*cttc)/(ensa*ctic+encca*cttc));
    rsp=((ensa*cttc-encca*ctic)/(ensa*cttc+encca*ctic));
    rsp=(q*q+rsp*rsp)/2;
    tsp=1-rsp;
}
}

else /* ptx is 0, horizontal, or undefined */
{
    q=((ensa*ctic-encca*cttc)/(ensa*ctic+encca*cttc));
    rsp=q*q;
    tsp=1-rsp;
}
}

/* tvsr is defined as tx ant height above receiver ant height */
tvsr= mymax(0.0,prop.tgh+prop.tsgh-prop.rch[1]);

if (d1a<50.0)
{
    arte=0.0195*crpc-20*log10(tsp);
}

else
{
    if (d1a<225.0)
    {

        if (tvsr>1000.0)
        {
            q=d1a*(0.03*exp(-0.14*pdk));
        }
        else
        {
            q=d1a*(0.07*exp(-0.17*pdk));
        }
    }
}

```

```

arte=q+(0.7*pdk-mymax(0.01,log10(prop.wn*47.7)-2))*(prop.hg[1]/hone);
}

else
{
q=0.00055*(pdk)+log10(pdk)*(0.041-0.0017*sqrt(hone)+0.019);

arte=d1a*q-(18*log10(rsp))/(exp(hone/37.5));

zi=1.5*sqrt(hone-prop.cch);

if(pdk>zi)
{
q=(pdk-zi)*10.2*((sqrt(mymax(0.01,log10(prop.wn*47.7)-2.0)))/(100-zi));
}
else
{
q=((zi-pdk)/zi)*(-20.0*mymax(0.01,log10(prop.wn*47.7)-2.0))/sqrt(hone);
}
arte=arte+q;

}

}

}

else /* for TX at or below clutter height */
{
q=(prop.cch-prop.tgh)*(2.06943-1.56184*exp(1/prop.cch-prop.tgh));
q=q+(17.98-0.84224*(prop.cch-prop.tgh))*exp(-0.00000061*pd);
arte=q+1.34795*20*log10(pd+1.0);
arte=arte-(mymax(0.01,log10(prop.wn*47.7)-2))*(prop.hg[1]/prop.tgh);
}

saalosv=arte;
/* propa.test0=saalosv; */

}

return saalosv;
}

```

```

double adiff(double d, prop_type &prop, propa_type &propa)
{
    complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
    static double wd1, xd1, afo, qk, aht, xht;
    double a, q, pk, ds, th, wa, ar, wd, adiffv;

    if (d==0)
    {
        q=prop.hg[0]*prop.hg[1];
        qk=prop.he[0]*prop.he[1]-q;

        if (prop.mdp<0.0)
            q+=10.0;

        wd1=sqrt(1.0+qk/q);
        xd1=propa.dla+propa.tha/prop.gme;
        q=(1.0-0.8*exp(-propa.dlsa/50e3))*prop.dh;
        q*=0.78*exp(-pow(q/16.0,0.25));
        afo=mymin(15.0,2.171*log(1.0+4.77e-4*prop.hg[0]*prop.hg[1]*prop.wn*q));
        qk=1.0/abs(prop_zgnd);
        aht=20.0;
        xht=0.0;

        for (int j=0; j<2; ++j)
        {
            /* a=0.5*pow(prop.dl[j],2.0)/prop.he[j]; */
            a=0.5*(prop.dl[j]*prop.dl[j])/prop.he[j];
            wa=pow(a*prop.wn,THIRD);
            pk=qk/wa;
            q=(1.607-pk)*151.0*wa*prop.dl[j]/a;
            xht+=q;
            aht+=fht(q,pk);
        }

        adiffv=0.0;
    }

    else

```

```

{
th=propa.tha+d*prop.gme;
ds=d-propa.dla;
/* q=0.0795775*prop.wn*ds*pow(th,2.0); */
q=0.0795775*prop.wn*ds*th*th;
adiffv=aknfe(q*prop.dl[0]/(ds+prop.dl[0]))+aknfe(q*prop.dl[1]/(ds+prop.dl[1]));
a=ds/th;
wa=pow(a*prop.wn,THIRD);
pk=qk/wa;
q=(1.607-pk)*151.0*wa*th+xht;
ar=0.05751*q-4.343*log(q)-aht;
q=(wd1+xd1/d)*mymin(((1.0-0.8*exp(-d/50e3))*prop.dh*prop.wn),6283.2);
wd=25.1/(25.1+sqrt(q));
adiffv=ar*wd+(1.0-wd)*adiffv+afo;
}

return adiffv;
}

```

```

double adiff2(double d, prop_type &prop, propa_type &propa)
{
complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
static double wd1, xd1, qk, aht, xht, toh, toho, roh, roho, dto, dto1, dtro, dro,
dro2, drto, dtr, dhh1, dhh2, dhec, dtot, dto1f, drof, dro2f;
double a, q, pk, rd, ds, dsl, dfdh, th, wa, ar, wd, sf1, sf2, ec, vv, kedr=0.0, arp=0.0,
sdr=0.0, pd=0.0, srp=0.0, kem=0.0, csd=0.0, sdl=0.0, adiffv2=0.0, closs=0.0;

sf1=1.0; /* average empirical hilltop foliage scatter factor for 1 obstruction */
sf2=1.0; /* average empirical hilltop foliage scatter factor for 2 obstructions */

dfdh=prop.dh;
ec=0.5*prop.gme;

/* adiff2 must first be run with d==0.0 to set up coefficients */
if (d==0)
{
q=prop.hg[0]*prop.hg[1];
qk=prop.he[0]*prop.he[1]-q;
dhec=2.73;

```

```

if (prop.mdp<0.0)
q+=10.0;

/* coefficients for a standard four radii, rounded earth computation are prepared */
wd1=sqrt(1.0+qk/q);
xd1=propa.dla+propa.tha/prop.gme;
q=(1.0-0.8*exp(-propa.dlsa/50e3))*prop.dh;
q*=0.78*exp(-pow(q/16.0,0.25));
qk=1.0/abs(prop_zgnd);
aht=20.0;
xht=0.0;
a=0.5*(prop.dl[0]*prop.dl[0])/prop.he[0];
wa=pow(a*prop.wn,THIRD);
pk=qk/wa;
q=(1.607-pk)*151.0*wa*prop.dl[0]/a;
xht=q;
aht+=fht(q,pk);

if ((int(prop.dl[1])==0.0) || (prop.the[1]>0.2))
{
    xht+=xht;
    aht+=(aht-20.0);
}

else
{
    a=0.5*(prop.dl[1]*prop.dl[1])/prop.he[1];
    wa=pow(a*prop.wn,THIRD);
    pk=qk/wa;
    q=(1.607-pk)*151.0*wa*prop.dl[1]/a;
    xht+=q;
    aht+=fht(q,pk);
}
adiffv2=0.0;
}

else

```

```

{
th=propa.tha+d*prop.gme;

dsl=mymax(d-propa.dla,0.0);
ds=d-propa.dla;
a=ds/th;
wa=pow(a*prop.wn,THIRD);
pk=qk/wa;
toh=prop.hht-(prop.rch[0]-prop.dl[0]*((prop.rch[1]-prop.rch[0])/prop.dist));
roh=prop.hhr-(prop.rch[0]-(prop.dist-prop.dl[1])*((prop.rch[1]-prop.rch[0])/prop.dist));
toho=prop.hht-(prop.rch[0]-(prop.dl[0]+dsl)*((prop.hhr-prop.rch[0])/(prop.dist-prop.dl[1])));
roho=prop.hhr-(prop.hht-dsl*((prop.rch[1]-prop.hht)/dsl));
dto=sqrt(prop.dl[0]*prop.dl[0]+toh*toh);
dto+=prop.gme*prop.dl[0];
dto1=sqrt(prop.dl[0]*prop.dl[0]+toho*toho);
dto1+=prop.gme*prop.dl[0];
dtro=sqrt((prop.dl[0]+dsl)*(prop.dl[0]+dsl)+prop.hhr*prop.hhr);
dtro+=prop.gme*(prop.dl[0]+dsl);
drto=sqrt((prop.dl[1]+dsl)*(prop.dl[1]+dsl)+prop.hht*prop.hht);
drto+=prop.gme*(prop.dl[1]+dsl);
dro=sqrt(prop.dl[1]*prop.dl[1]+roh*roh);
dro+=prop.gme*(prop.dl[1]);
dro2=sqrt(prop.dl[1]*prop.dl[1]+roho*roho);
dro2+=prop.gme*(prop.dl[1]);
dtr=sqrt(prop.dist*prop.dist+(prop.rch[0]-prop.rch[1])*(prop.rch[0]-prop.rch[1]));
dtr+=prop.gme*prop.dist;
dhh1=sqrt((prop.dist-propa.dla)*(prop.dist-propa.dla)+toho*toho);
dhh1+=prop.gme*(prop.dist-propa.dla);
dhh2=sqrt((prop.dist-propa.dla)*(prop.dist-propa.dla)+roho*roho);
dhh2+=prop.gme*(prop.dist-propa.dla);

/* for 1 obst tree base path */
dtof=sqrt(prop.dl[0]*prop.dl[0]+(toh-prop.cch)*(toh-prop.cch));
dtof+=prop.gme*prop.dl[0];
dto1f=sqrt(prop.dl[0]*prop.dl[0]+(toho-prop.cch)*(toho-prop.cch));
dto1f+=prop.gme*prop.dl[0];
drof=sqrt(prop.dl[1]*prop.dl[1]+(roh-prop.cch)*(roh-prop.cch));
drof+=prop.gme*(prop.dl[1]);
dro2f=sqrt(prop.dl[1]*prop.dl[1]+(roho-prop.cch)*(roho-prop.cch));

```

```

dro2f+=prop.gme*(prop.dl[1]);

/* saalos coefficients preset for post-obstacle receive path */
prop.tgh=prop.cch+1.0;
prop.tsgh=prop.hhr;
rd=prop.dl[1];

/* two obstacle diffraction calculation */
if (int(ds)>0) /* there are 2 obstacles */
{
    if(int(prop.dl[1])>0.0) /* receive site past 2nd peak */
    {
        /* rounding attenuation */
        q=(1.607-pk)*151.0*wa*th+xht;
        ar=0.05751*q-10*log10(q)-aht;

        /* knife edge vs round weighting */
        q=(1.0-0.8*exp(-d/50e3))*prop.dh;
        q=(wd1+xd1/d)*mymin((q*prop.wn),6283.2);
        wd=25.1/(25.1+sqrt(q));

        q=0.6365*prop.wn;

        if(prop.the[1]<0.2) /* receive grazing angle below 0.2 rad */
        {
            /* knife edge attenuation for two obstructions */

            if(prop.hht < 3400) /* if below tree line, foliage top loss */
            {
                vv=q*abs(dto1+dhh1-dtro);
                adiffv2=-18.0+sf2*aknfe(vv);
            }
            else
            {
                vv=q*abs(dto1+dhh1-dtro);
                adiffv2=aknfe(vv);
            }
        }

        if(prop.hhr < 3400)
    }
}

```

```

{
  vv=q*abs(dro2+dhh2-drto);
  adiffv2+=(-18.0+sf2*aknfe(vv));
}
else
{
  vv=q*abs(dro2+dhh2-drto);
  adiffv2+=aknfe(vv);
}
/* finally, add clutter loss */
closs=saalos(rd, prop, propa);
adiffv2+=mymin(22.0,closs);

}

else /* rcvr site too close to 2nd obs */
{
/* knife edge attenuation for 1st obs */

if(prop.hht < 3400)
{
  vv=q*abs(dto1+dhh1-dtro);
  adiffv2=-18.0+sf2*aknfe(vv);
}
else
{
  vv=q*abs(dto1+dhh1-dtro);
  adiffv2=aknfe(vv);
}

/* weighted calc. of knife vs rounded edge
adiffv2=ar*wd+(1.0-wd)*adiffv2; */

/* clutter path loss past 2nd peak */
if(prop.the[1]<1.22)
{
  rd=prop.dl[1];

  if(prop.the[1]>0.6) /* through foliage downhill */
  {

```

```

prop.tgh=prop.cch;
}
else /* close to foliage, rcvr in foliage downslope */
{
vv=0.6365*prop.wn*abs(dro2+dhh2-drto);
}
adiffv2+=aknfe(vv);
closs=saalos(rd, prop, propa);
adiffv2+=mymin(closs,22.0);
}
else /* rcvr very close to bare cliff or skyscraper */
{
adiffv2=5.8+25.0;
}
}
}
}
else /* receive site is atop a 2nd peak */
{
vv=0.6365*prop.wn*abs(dto+dro-dtr);
adiffv2=5.8 + aknfe(vv);
}
}
else /* for single obstacle */
{
if(int(prop.dl[1])>0.0) /* receive site past 1st peak */
{
if(prop.the[1]<0.2) /* receive grazing angle less than .2 radians */
{
vv=0.6365*prop.wn*abs(dto+dro-dtr);

if(prop.hht < 3400)
{
sdl=18.0;
sdl=pow(10,(-sdl/20));
/* ke phase difference with respect to direct t-r line */
kedr=0.159155*prop.wn*abs(dto+dro-dtr);
arp=abs(kedr-(int(kedr)));

```

```

kem=aknfe(vv);
kem= pow(10,(-kem/20));
/* scatter path phase with respect to direct t-r line */
sdr=0.5+0.159155*prop.wn*abs(dtot+drof-dtr);
srp=abs(sdr-(int(sdr)));
/* difference between scatter and ke phase in radians */
pd=6.283185307*abs(srp-arp);
/* report pd prior to restriction */
propa.test4=pd;
/* keep pd between 0 and pi radians and adjust for 3&4 quadrant */
if(pd>=3.141592654)
{
    pd=6.283185307-pd;
    csd=abq_alos(complex<double>(sdl,0)+complex<double>(kem*-cos(pd), kem*-sin(pd)));
}
else
{
    csd=abq_alos(complex<double>(sdl,0)+complex<double>(kem*cos(pd), kem*sin(pd)));
}
/*csd=mymax(csd,0.0009); limits maximum loss value to 30.45 db */
adiffv2=-3.71-10*log10(csd);
}
else
{
    adiffv2=aknfe(vv);
}
/* finally, add clutter loss */
closs=saalos(rd, prop, propa);
adiffv2+=mymin(closs,22.0);
}
else /* receive grazing angle too high */
{
    if(prop.the[1]<1.22)
    {
        rd=prop.dl[1];
        if(prop.the[1]>0.6) /* through foliage downhill */
        {
            prop.tgh=prop.cch;
        }
    }
}

```

```

    }
else /* downhill slope just above foliage */
{
    vv=0.6365*prop.wn*abs(dto+dro-dtr);
    adiffv2=aknfe(vv);
}
closs=saalos(rd, prop, propa);
adiffv2+=mymin(22.0,closs);
}
else /* receiver very close to bare cliff or skyscraper */
{
    adiffv2=5.8+25.0;
}
}
}
}
else /* if occurs, receive site atop first peak */
{
    adiffv2=5.8;
}
}
}
}
propa.test0=adiffv2;
propa.test1=dro;
propa.test2=kem;
propa.test3=csd;

return adiffv2;
}

```

```

double ascat( double d, prop_type &prop, propa_type &propa)
{
    static double ad, rr, etq, h0s;
    double h0, r1, r2, z0, ss, et, ett, th, q;
    double ascatv, temp;

    if (d==0.0)
    {
        ad=prop.dl[0]-prop.dl[1];
        rr=prop.he[1]/prop.rch[0];

```

```

if (ad<0.0)
{
    ad=-ad;
    rr=1.0/rr;
}

etq=(5.67e-6*prop.ens-2.32e-3)*prop.ens+0.031;
h0s=-15.0;
ascatv=0.0;
}

else
{
if (h0s>15.0)
    h0=h0s;
else
{
    th=prop.the[0]+prop.the[1]+d*prop.gme;
    r2=2.0*prop.wn*th;
    r1=r2*prop.he[0];
    r2*=prop.he[1];

    if (r1<0.2 && r2<0.2)
        return 1001.0; // ===== early return

    ss=(d-ad)/(d+ad);
    q=rr/ss;
    ss=mymax(0.1,ss);
    q=mymin(mymax(0.1,q),10.0);
    z0=(d-ad)*(d+ad)*th*0.25/d;
/* et=(etq*exp(-pow(mymin(1.7,z0/8.0e3),6.0))+1.0)*z0/1.7556e3; */

    temp=mymin(1.7,z0/8.0e3);
    temp=temp*temp*temp*temp*temp*temp;
    et=(etq*exp(-temp)+1.0)*z0/1.7556e3;

    ett=mymax(et,1.0);
    h0=(h0f(r1,ett)+h0f(r2,ett))*0.5;
}
}

```

```

h0+=mymin(h0,(1.38-log(ett))*log(ss)*log(q)*0.49);
h0=FORTRAN_DIM(h0,0.0);

if (et<1.0)
{
/* h0=et*h0+(1.0-et)*4.343*log(pow((1.0+1.4142/r1)*(1.0+1.4142/r2),2.0)*(r1+r2)/(r1+r2+2.8284)); */

temp=((1.0+1.4142/r1)*(1.0+1.4142/r2));
h0=et*h0+(1.0-et)*4.343*log((temp*temp)*(r1+r2)/(r1+r2+2.8284));
}

if (h0>15.0 && h0s>=0.0)
h0=h0s;
}

h0s=h0;
th=propa.tha+d*prop.gme;
/* ascatv=ahd(th*d)+4.343*log(47.7*prop.wn*pow(th,4.0))-0.1*(prop.ens-301.0)*exp(-th*d/40e3)+h0; */
ascatv=ahd(th*d)+4.343*log(47.7*prop.wn*(th*th*th*th))-0.1*(prop.ens-301.0)*exp(-th*d/40e3)+h0;
}

return ascatv;
}

double qerfi(double q)
{
double x, t, v;
double c0=2.515516698;
double c1=0.802853;
double c2=0.010328;
double d1=1.432788;
double d2=0.189269;
double d3=0.001308;

x=0.5-q;
t=mymax(0.5-fabs(x),0.000001);
t=sqrt(-2.0*log(t));
v=t-((c2*t+c1)*t+c0)/(((d3*t+d2)*t+d1)*t+1.0);

```

```

if (x<0.0)
v=-v;

return v;
}

void qlrps(double fmhz, double zsys, double en0, int ipol, double eps, double sgm, prop_type &prop)
{
double gma=157e-9;

prop.wn=fmhz/47.7;
prop.ens=en0;

if (zsys!=0.0)
prop.ens*=exp(-zsys/9460.0);

prop.gme=gma*(1.0-0.04665*exp(prop.ens/179.3));
complex<double> zq, prop_zgnd(prop.zgndreal,prop.zgndimag);
zq=complex<double> (eps,376.62*sgm/prop.wn);
prop_zgnd=sqrt(zq-1.0);

if (ipol!=0.0)
prop_zgnd=prop_zgnd/zq;

prop.zgndreal=prop_zgnd.real();
prop.zgndimag=prop_zgnd.imag();

}

double alos(double d, prop_type &prop, propa_type &propa)
{
complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
static double wls;
complex<double> r;
double s, sps, q;
double alosv;

if (d==0.0)
{

```

```

wls=0.021/(0.021+prop.wn*prop.dh/mymax(10e3,propa.dlsa));
alosv=0.0;
}

else
{
q=(1.0-0.8*exp(-d/50e3))*prop.dh;
s=0.78*q*exp(-pow(q/16.0,0.25));
q=prop.he[0]+prop.he[1];
sps=q/sqrt(d*d+q*q);
r=(sps-prop_zgnd)/(sps+prop_zgnd)*exp(-mymin(10.0,prop.wn*s*sps));
q=abq_alos(r);

if (q<0.25 || q<sps)
r=r*sqrt(sps/q);

alosv=propa.emd*d+propa.aed;
q=prop.wn*prop.he[0]*prop.he[1]*2.0/d;

if (q>1.57)
q=3.14-2.4649/q;

alosv=(-4.343*log(abq_alos(complex<double>(cos(q),-sin(q))+r))-alosv)*wls+alosv;

}
return alosv;
}

```

```

double alos2(double d, prop_type &prop, propa_type &propa)
{
complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
complex<double> r;
double cd, cr, dr, hr, hrg, ht, htg, hrp, re, s, sps, q, pd, drh;
int rp;
double alosv;

cd=0.0;
cr=0.0;

```

```

htg=prop.hg[0];
hrg=prop.hg[1];
ht=prop.ght;
hr=prop.ghr;
rp=prop.rpl;
hrp=prop.rph;
pd=prop.dist;

if (d==0.0)
{
    alosv=0.0;
}

else
{
    q=prop.he[0]+prop.he[1];
    sps=q/sqrt(pd*pd+q*q);
    q=(1.0-0.8*exp(-pd/50e3))*prop.dh;

    if (prop.mdp<0)
    {
        dr=pd/(1+hrg/htg);

        if (dr<(0.5*pd))
        {
            drh=6378137.0-sqrt(-(0.5*pd)*(0.5*pd)+6378137.0*6378137.0+(0.5*pd-dr)*(0.5*pd-dr));
        }
        else
        {
            drh=6378137.0-sqrt(-(0.5*pd)*(0.5*pd)+6378137.0*6378137.0+(dr-0.5*pd)*(dr-0.5*pd));
        }
    }

    if ((sps<0.05) && (prop.cch>hrg) && (prop.dist< prop.dl[0])) /* if far from transmitter and receiver
below canopy */
    {
        cd=mymax(0.01,pd*(prop.cch-hrg)/(htg-hrg));
        cr=mymax(0.01,pd-dr+dr*(prop.cch-drh)/htg);
        q=((1.0-0.8*exp(-pd/50e3))*prop.dh*(mymin(-20*log10(cd/cr),1.0)));
    }
}

```

```

}

s=0.78*q*exp(-pow(q/16.0,0.25));
q=exp(-mymin(10.0,prop.wn*s*sps));
r=q*(sps-prop_zgnd)/(sps+prop_zgnd);
q=abq_alos(r);
q=mymin(q,1.0);

if (q<0.25 || q<sps)
{
  r=r*sqrt(sps/q);
}
q=prop.wn*prop.he[0]*prop.he[1]/(pd*3.1415926535897);

if (prop.mdp<0)
{
  q=prop.wn*((ht-hrp)*(hr-hrp))/(pd*3.1415926535897);
}
q=floor(q);

if (q<0.5)
{
  q*=3.1415926535897;
}

else
{
  q=(1-q)*3.1415926535897;
}

/* no longer valid complex conjugate removed
   by removing minus sign from in front of sin function */
re=abq_alos(complex<double>(cos(q),sin(q))+r);
alosv=-10*log10(re);
prop.tgh=prop.hg[0]; /*tx above gnd hgt set to antenna height AGL */
prop.tsgh=prop.rch[0]-prop.hg[0]; /* tsgh set to tx site gl AMSL */

if ((prop.hg[1]<prop.cch) && (prop.thera<0.785) && (prop.thenr<0.785))
{
  if (sps<0.05)

```

```

{
    alosv=alosv+saalos(pd, prop, propa);
}
else
{
    alosv=saalos(pd, prop, propa);
}
}
}

/*propa.test1=alosv; */
alosv = mymin(22.0,alosv);
return alosv;
}

```

```

void qlra(int kst[], int klimx, int mdvarx, prop_type &prop, propv_type &propv)
{
double q;

for (int j=0; j<2; ++j)
{
if (kst[j]<=0)
    prop.he[j]=prop.hg[j];
else
{
    q=4.0;

if (kst[j]!=1)
    q=9.0;

if (prop.hg[j]<5.0)
    q*=sin(0.3141593*prop.hg[j]);

prop.he[j]=prop.hg[j]+(1.0+q)*exp(-mymin(20.0,2.0*prop.hg[j]/mymax(1e-3,prop.dh)));
}

q=sqrt(2.0*prop.he[j]/prop.gme);
prop.dl[j]=q*exp(-0.07*sqrt(prop.dh/mymax(prop.he[j],5.0)));
prop.the[j]=(0.65*prop.dh*(q/prop.dl[j]-1.0)-2.0*prop.he[j])/q;
}

```

```

prop.mdp=1;
propv.lvar=mymax(propv.lvar,3);

if (mdvarx>=0)
{
    propv.mdvar=mdvarx;
    propv.lvar=mymax(propv.lvar,4);
}

if (klimx>0)
{
    propv.klim=klimx;
    propv.lvar=5;
}
}

```

```

void Irprop (double d, prop_type &prop, propa_type &propa)
{
/* PaulM_Irprop used for ITM */
static bool wlos, wscat;
static double dmin, xae;
complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
double a0, a1, a2, a3, a4, a5, a6;
double d0, d1, d2, d3, d4, d5, d6;
bool wq;
double q;
int j;

if (prop.mdp!=0)
{
    for (j=0; j<2; j++)
        propa.dls[j]=sqrt(2.0*prop.he[j]/prop.gme);

    propa.dlsa=propa.dls[0]+propa.dls[1];
    propa.dla=prop.dl[0]+prop.dl[1];
    propa.tha=mymax(prop.the[0]+prop.the[1],-propa.dla*prop.gme);
    wlos=false;
}

```

```

wscat=false;

if (prop.wn<0.838 || prop.wn>210.0)
prop.kwx=mymax(prop.kwx,1);

for (j=0; j<2; j++)
if (prop.hg[j]<1.0 || prop.hg[j]>1000.0)
prop.kwx=mymax(prop.kwx,1);

for (j=0; j<2; j++)
if (abs(prop.the[j]) >200e-3 || prop.dl[j]<0.1*propa.dls[j] || prop.dl[j]>3.0*propa.dls[j] )
prop.kwx=mymax(prop.kwx,3);

if (prop.ens < 250.0 || prop.ens > 400.0 || prop.gme < 75e-9 || prop.gme > 250e-9 || prop_zgnd.real()
<= abs(prop_zgnd.imag()) || prop.wn < 0.419 || prop.wn > 420.0)
prop.kwx=4;

for (j=0; j<2; j++)
if (prop.hg[j]<0.5 || prop.hg[j]>3000.0)
prop.kwx=4;

dmin=abs(prop.he[0]-prop.he[1])/200e-3;
q=adiff(0.0,prop,propa);
/* xae=pow(prop.wn*pow(prop.gme,2.),-THIRD); -- JDM made argument 2 a double */
xae=pow(prop.wn*(prop.gme*prop.gme),-THIRD); /* No 2nd pow() */
d3=mymax(propa.dlsa,1.3787*xae+propa.dla);
d4=d3+2.7574*xae;
a3=adiff(d3,prop,propa);
a4=adiff(d4,prop,propa);
propa.emd=(a4-a3)/(d4-d3);
propa.aed=a3-propa.emd*d3;
}

if (prop.mdp>=0)
{
prop.mdp=0;
prop.dist=d;
}

```

```

if (prop.dist>0.0)
{
if (prop.dist>1000e3)
prop.kwx=mymax(prop.kwx,1);

if (prop.dist<dmin)
prop.kwx=mymax(prop.kwx,3);

if (prop.dist<1e3 || prop.dist>2000e3)
prop.kwx=4;
}

if (prop.dist<propa.dlsa)
{
if (!wlos)
{
q=alos(0.0,prop,propa);
d2=propa.dlsa;
a2=propa.aed+d2*propa.emd;
d0=1.908*prop.wn*prop.he[0]*prop.he[1];

if (propa.aed>=0.0)
{
d0=mymin(d0,0.5*propa.dla);
d1=d0+0.25*(propa.dla-d0);
}
}

else
d1=mymax(-propa.aed/propa.emd,0.25*propa.dla);

a1=alos(d1,prop,propa);
wq=false;

if (d0<d1)
{
a0=alos(d0,prop,propa);
q=log(d2/d0);
propa.ak2=mymax(0.0,((d2-d0)*(a1-a0)-(d1-d0)*(a2-a0))/((d2-d0)*log(d1/d0)-(d1-d0)*q));
wq=propa.aed>=0.0 || propa.ak2>0.0;
}
}

```

```

if (wq)
{
    propa.ak1=(a2-a0-propa.ak2*q)/(d2-d0);

    if (propa.ak1<0.0)
    {
        propa.ak1=0.0;
        propa.ak2=FORTRAN_DIM(a2,a0)/q;

        if (propa.ak2==0.0)
            propa.ak1=propa.emd;
    }
}

else
{
    propa.ak2=0.0;
    propa.ak1=(a2-a1)/(d2-d1);

    if (propa.ak1<=0.0)
        propa.ak1=propa.emd;
}
}

else
{
    propa.ak1=(a2-a1)/(d2-d1);
    propa.ak2=0.0;

    if (propa.ak1<=0.0)
        propa.ak1=propa.emd;
}

propa.ael=a2-propa.ak1*d2-propa.ak2*log(d2);
wlos=true;
}

if (prop.dist>0.0)

```

```

prop.aref=propa.ael+propa.ak1*prop.dist+propa.ak2*log(prop.dist);

}

if (prop.dist<=0.0 || prop.dist>=propa.dlsa)
{
if(!wscat)
{
q=ascat(0.0,prop,propa);
d5=propa.dla+200e3;
d6=d5+200e3;
a6=ascat(d6,prop,propa);
a5=ascat(d5,prop,propa);

if (a5<1000.0)
{
propa.ems=(a6-a5)/200e3;
propa.dx=mymax(propa.dlsa,mymax(propa.dla+0.3*xae*log(47.7*prop.wn),(a5-propa.aed-
propa.ems*d5)/(propa.emd-propa.ems)));
propa.aes=(propa.emd-propa.ems)*propa.dx+propa.aed;
}

else
{
propa.ems=propa.emd;
propa.aes=propa.aed;
propa.dx=10.e6;
}

wscat=true;
}

if (prop.dist>propa.dx)
prop.aref=propa.aes+propa.ems*prop.dist;
else
prop.aref=propa.aed+propa.emd*prop.dist;
}

prop.aref=mymax(prop.aref,0.0);

```

```
}
```

```
void lrprop2(double d, prop_type &prop, propa_type &propa)
{
/* ITWOM_Lrprop2 */
static bool wlos, wscat;
static double dmin, xae;
complex<double> prop_zgnd(prop.zgndreal,prop.zgndimag);
double pd1;
double a0, a1, a2, a3, a4, a5, a6, iw;
double d0, d1, d2, d3, d4, d5, d6;
bool wq;
double q;
int j;

iw=prop.tiw;
pd1=prop.dist;
propa.dx=2000000.0;

if (prop.mdp!=0) /* if oper. mode is not 0, i.e. not area mode ongoing */
{
for (j=0; j<2; j++)
propa.dls[j]=sqrt(2.0*prop.he[j]/prop.gme);

propa.dlsa=propa.dls[0]+propa.dls[1];
propa.dlsa=mymin(propa.dlsa,1000000.0);
propa.dla=prop.dl[0]+prop.dl[1];
propa.tha=mymax(prop.the[0]+prop.the[1],-propa.dla*prop.gme);
wlos=false;
wscat=false;

/*checking for parameters-in-range, error codes set if not */

if (prop.wn<0.838 || prop.wn>210.0)
prop.kwx=mymax(prop.kwx,1);
```

```

for (j=0; j<2; j++)
if (prop.hg[j]<1.0 || prop.hg[j]>1000.0)
prop.kwx=mymax(prop.kwx,1);

if(abs(prop.the[0])>200e-3)
prop.kwx=mymax(prop.kwx,3);

if(abs(prop.the[1])>1.220)
prop.kwx=mymax(prop.kwx,3);

/*for (j=0; j<2; j++)
if (prop.dl[j]<0.1*propa.dls[j] || prop.dl[j]>3.0*propa.dls[j])
prop.kwx=mymax(prop.kwx,3); */

if (prop.ens<250.0 || prop.ens>400.0 || prop.gme<75e-9 || prop.gme>250e-9 || prop_zgnd.real()
<=abs(prop_zgnd.imag()) || prop.wn<0.419 || prop.wn>420.0)
prop.kwx=4;

for (j=0; j<2; j++)

if (prop.hg[j]<0.5 || prop.hg[j]>3000.0)
prop.kwx=4;

dmin=abs(prop.he[0]-prop.he[1])/200e-3;
q=adiff2(0.0,prop,propa);
xae=pow(prop.wn*(prop.gme*prop.gme),-THIRD);
d3=mymax(propa.dlsa,1.3787*xae+propa.dla);
d4=d3+2.7574*xae;
a3=adiff2(d3,prop,propa);
a4=adiff2(d4,prop,propa);
propa.emd=(a4-a3)/(d4-d3);
propa.aed=a3-propa.emd*d3;
}

if (prop.mdp>=0) /* if initializing the area mode */
{
prop.mdp=0; /* area mode is initialized */
prop.dist=d;
}

```

```

if (prop.dist>0.0)
{
if (prop.dist>1000e3) /* prop.dist being in meters, if greater than 1000 km, kwx=1 */
prop.kwx=mymax(prop.kwx,1);

if (prop.dist<dmin)
prop.kwx=mymax(prop.kwx,3);

if (prop.dist<1e3 || prop.dist>2000e3)
prop.kwx=4;
}

if (prop.dist<propa.dlsa)
{

if (iw<=0.0) /* if interval width is zero or less, used for area mode */
{
if (!wlos)
{
q=alos2(0.0,prop,propa);
d2=propa.dlsa;
a2=propa.aed+d2*propa.emd;
d0=1.908*prop.wn*prop.he[0]*prop.he[1];

if (propa.aed>0.0)
{
prop.aref=propa.aed+propa.emd*prop.dist;
}
else
{
if (propa.aed==0.0)
{
d0=mymin(d0,0.5*propa.dla);
d1=d0+0.25*(propa.dla-d0);
}
else /* aed less than zero */
{

```

```

d1=mymax(-propa.aed/propa.emd,0.25*propa.dla);
}
a1=alos2(d1,prop,propa);
wq=false;

if (d0<d1)
{
a0=alos2(d0,prop,propa);
a2=mymin(a2,alos2(d2,prop,propa));
q=log(d2/d0);
propa.ak2=mymax(0.0,((d2-d0)*(a1-a0)-(d1-d0)*(a2-a0))/((d2-d0)*log(d1/d0)-(d1-d0)*q));
wq=propa.aed>=0.0 || propa.ak2>0.0;

if (wq)
{
propa.ak1=(a2-a0-propa.ak2*q)/(d2-d0);

if (propa.ak1<0.0)
{
propa.ak1=0.0;
propa.ak2=FORTRAN_DIM(a2,a0)/q;

if (propa.ak2==0.0)
propa.ak1=propa.emd;
}

}

if(!wq)
{
propa.ak1=FORTRAN_DIM(a2,a1)/(d2-d1);
propa.ak2=0.0;

if (propa.ak1==0.0)
propa.ak1=propa.emd;

}

propa.ael=a2-propa.ak1*d2-propa.ak2*log(d2);
wlos=true;

```

```

    }
}

}

else /* for ITWOM point-to-point mode */
{
    if (!wlos)
    {
        q=alos2(0.0,prop,propa); /* coefficient setup */
        wlos=true;
    }

    if (prop.los==1) /* if line of sight */
    {
        prop.aref=alos2(pd1,prop,propa);
    }
    else
    {
        if (int(prop.dist-prop.dl[0])==0) /* if at 1st horiz */
        {
            prop.aref=5.8+alos2(pd1,prop,propa);
        }
        else if (int(prop.dist-prop.dl[0])>0.0) /* if past 1st horiz */
        {
            q=adiff2(0.0,prop,propa);
            prop.aref=adiff2(pd1,prop,propa);
        }
        else
        {
            prop.aref=1.0;
        }
    }
}
}

/* los and diff. range coefficents done. Starting troposcatter */
if (prop.dist<=0.0 || prop.dist>=propa.dlsa)
{

```

```

if (iw==0.0) /* area mode */
{
if(!wscat)
{
q=ascat(0.0,prop,propa);
d5=propa.dla+200e3;
d6=d5+200e3;
a6=ascat(d6,prop,propa);
a5=ascat(d5,prop,propa);

if (a5<1000.0)
{
propa.ems=(a6-a5)/200e3;
propa.dx=mymax(propa.dlsa,mymax(propa.dla+0.3*xae*log(47.7*prop.wn),(a5-propa.aed-
propa.ems*d5)/(propa.emd-propa.ems)));

propa.aes=(propa.emd-propa.ems)*propa.dx+propa.aed;
}

else
{
propa.ems=propa.emd;
propa.aes=propa.aed;
propa.dx=10000000;
}
wscat=true;
}

if (prop.dist>propa.dx)
{
prop.aref=propa.aes+propa.ems*prop.dist;
}
else
{
prop.aref=propa.aed+propa.emd*prop.dist;
}
}

else /* ITWOM mode q used to preset coefficients with zero input */
{

```

```

if(!wscat)
{
d5=0.0;
d6=0.0;
q=ascat(0.0,prop,propa);
a6=ascat(pd1,prop,propa);
q=adiff2(0.0,prop,propa);
a5=adiff2(pd1,prop,propa);

if (a5<=a6)
{
propa.dx=10000000;
prop.oref=a5;
}
else
{
propa.dx=propa.dlsa;
prop.oref=a6;
}
wscat=true;
}
}
}

prop.oref=mymax(prop.oref,0.0);
}

```

```

double curve (double const &c1, double const &c2, double const &x1,
              double const &x2, double const &x3, double const &de)
{
/* return (c1+c2/(1.0+pow((de-x2)/x3,2.0)))*pow(de/x1,2.0)/(1.0+pow(de/x1,2.0)); */
double temp1, temp2;

temp1=(de-x2)/x3;
temp2=de/x1;

temp1*=temp1;
temp2*=temp2;

```

```

return (c1+c2/(1.0+temp1))*temp2/(1.0+temp2);
}

double avar(double zzt, double zzl, double zzc, prop_type &prop, propv_type &propv)
{
    static int kdv;
    static double dexa, de, vmd, vs0, sgl, sgtn, sgtp, sgtd, tgtd,
    gm, gp, cv1, cv2, yv1, yv2, yv3, csm1, csm2, ysm1, ysm2,
    ysm3, csp1, csp2, ysp1, ysp2, ysp3, csd1, zd, cfm1, cfm2,
    cfm3, cfp1, cfp2, cfp3;

    double bv1[7]={-9.67,-0.62,1.26,-9.21,-0.62,-0.39,3.15};
    double bv2[7]={12.7,9.19,15.5,9.05,9.19,2.86,857.9};
    double xv1[7]={144.9e3,228.9e3,262.6e3,84.1e3,228.9e3,141.7e3,2222.e3};
    double xv2[7]={190.3e3,205.2e3,185.2e3,101.1e3,205.2e3,315.9e3,164.8e3};
    double xv3[7]={133.8e3,143.6e3,99.8e3,98.6e3,143.6e3,167.4e3,116.3e3};
    double bsm1[7]={2.13,2.66,6.11,1.98,2.68,6.86,8.51};
    double bsm2[7]={159.5,7.67,6.65,13.11,7.16,10.38,169.8};
    double xsm1[7]={762.2e3,100.4e3,138.2e3,139.1e3,93.7e3,187.8e3,609.8e3};
    double xsm2[7]={123.6e3,172.5e3,242.2e3,132.7e3,186.8e3,169.6e3,119.9e3};
    double xsm3[7]={94.5e3,136.4e3,178.6e3,193.5e3,133.5e3,108.9e3,106.6e3};
    double bsp1[7]={2.11,6.87,10.08,3.68,4.75,8.58,8.43};
    double bsp2[7]={102.3,15.53,9.60,159.3,8.12,13.97,8.19};
    double xsp1[7]={636.9e3,138.7e3,165.3e3,464.4e3,93.2e3,216.0e3,136.2e3};
    double xsp2[7]={134.8e3,143.7e3,225.7e3,93.1e3,135.9e3,152.0e3,188.5e3};
    double xsp3[7]={95.6e3,98.6e3,129.7e3,94.2e3,113.4e3,122.7e3,122.9e3};
    double bsd1[7]={1.224,0.801,1.380,1.000,1.224,1.518,1.518};
    double bzd1[7]={1.282,2.161,1.282,20.,1.282,1.282,1.282};
    double bfm1[7]={1.0,1.0,1.0,1.0,0.92,1.0,1.0};
    double bfm2[7]={0.0,0.0,0.0,0.0,0.25,0.0,0.0};
    double bfm3[7]={0.0,0.0,0.0,0.0,1.77,0.0,0.0};
    double bfp1[7]={1.0,0.93,1.0,0.93,0.93,1.0,1.0};
    double bfp2[7]={0.0,0.31,0.0,0.19,0.31,0.0,0.0};
    double bfp3[7]={0.0,2.00,0.0,1.79,2.00,0.0,0.0};

    static bool ws, w1;
    double rt=7.8, rl=24.0, avarv, q, vs, zt, zl, zc;
    double sgt, yr, temp1, temp2;
    int temp_klim=propv.klim-1;
}

```

```

if (propv.lvar>0)
{
switch (propv.lvar)
{
default:
if (propv.klim<=0 || propv.klim>7)
{
propv.klim=5;
temp_klim=4;
prop.kwx=mymax(prop.kwx,2);
}

cv1=bv1[temp_klim];
cv2=bv2[temp_klim];
yv1=xv1[temp_klim];
yv2=xv2[temp_klim];
yv3=xv3[temp_klim];
csm1=bsm1[temp_klim];
csm2=bsm2[temp_klim];
ysm1=xsm1[temp_klim];
ysm2=xsm2[temp_klim];
ysm3=xsm3[temp_klim];
csp1=bsp1[temp_klim];
csp2=bsp2[temp_klim];
ysp1=xsp1[temp_klim];
ysp2=xsp2[temp_klim];
ysp3=xsp3[temp_klim];
csd1=bsd1[temp_klim];
zd=bzd1[temp_klim];
cfm1=bfm1[temp_klim];
cfm2=bfm2[temp_klim];
cfm3=bfm3[temp_klim];
cfp1=bfp1[temp_klim];
cfp2=bfp2[temp_klim];
cfp3=bfp3[temp_klim];

```

case 4:

```

kdv=propv.mdvar;
ws=kdv>=20;
```

```

if (ws)
    kdv=20;

w1=kdv>=10;

if (w1)
    kdv=10;

if (kdv<0 || kdv>3)
{
    kdv=0;
    prop.kwx=mymax(prop.kwx,2);
}

case 3:
q=log(0.133*prop.wn);

/* gm=cfm1+cfm2/(pow(cfm3*q,2.0)+1.0); */
/* gp=cfp1+cfp2/(pow(cfp3*q,2.0)+1.0); */

gm=cfm1+cfm2/((cfm3*q*cfm3*q)+1.0);
gp=cfp1+cfp2/((cfp3*q*cfp3*q)+1.0);

case 2:
dexa=sqrt(18e6*prop.he[0])+sqrt(18e6*prop.he[1])+pow((575.7e12/prop.wn),THIRD);

case 1:
if (prop.dist<dixa)
    de=130e3*prop.dist/dixa;
else
    de=130e3+prop.dist-dixa;
}

vmd=curve(cv1,cv2,yv1,yv2,yv3,de);
sgtm=curve(csm1,csm2,ysm1,ysm2,ysm3,de)*gm;
sgtp=curve(csp1,csp2,ysp1,ysp2,ysp3,de)*gp;
sgtd=sgtp*csd1;
tgtd=(sgtp-sgtd)*zd;

```

```
if (w1)
    sgl=0.0;
else
{
    q=(1.0-0.8*exp(-prop.dist/50e3))*prop.dh*prop.wn;
    sgl=10.0*q/(q+13.0);
}
```

```
if (ws)
    vs0=0.0;
else
{
/* vs0=pow(5.0+3.0*exp(-de/100e3),2.0); */
temp1=(5.0+3.0*exp(-de/100e3));
vs0=temp1*temp1;
```

```
}
```

```
propv.lvar=0;
}
```

```
zt=zzt;
zl=zzl;
zc=zzc;
```

```
switch (kdv)
{
case 0:
zt=zc;
zl=zc;
break;
```

```
case 1:
zl=zc;
break;
```

```
case 2:
zl=zt;
```

```

}

if (fabs(zt)>3.1 || fabs(zl)>3.1 || fabs(zc)>3.1)
prop.kwx=mymax(prop.kwx,1);

if (zt<0.0)
sgt=sgtm;

else if (zt<=zd)
sgt=sgtp;

else
sgt=sgtd+tgtd/zt;

/* vs=vs0+pow(sgt*zt,2.0)/(rt+zc*zc)+pow(sgl*zl,2.0)/(rl+zc*zc); */

temp1=sgt*zt;
temp2=sgl*zl;

vs=vs0+(temp1*temp1)/(rt+zc*zc)+(temp2*temp2)/(rl+zc*zc);

if (kdv==0)
{
yr=0.0;
propv.sgc=sqrt(sgt*sgt+sgl*sgl+vs);
}

else if (kdv==1)
{
yr=sgt*zt;
propv.sgc=sqrt(sgl*sgl+vs);
}

else if (kdv==2)
{
yr=sqrt(sgt*sgt+sgl*sgl)*zt;
propv.sgc=sqrt(vs);
}

```

```

else
{
    yr=sgt*zt+sgl*zl;
    propv.sgc=sqrt(vs);
}

avarv=prop.aref-vmd-yr-propv.sgc*zc;

if (avarv<0.0)
    avarv=avarv*(29.0-avarv)/(29.0-10.0*avarv);

return avarv;
}

```

```

void hzns(double pfl[], prop_type &prop)
{
/* Used only with ITM 1.2.2 */
bool wq;
int np;
double xi, za, zb, qc, q, sb, sa;

np=(int)pfl[0];
xi=pfl[1];
za=pfl[2]+prop.hg[0];
zb=pfl[np+2]+prop.hg[1];
qc=0.5*prop.gme;
q=qc*prop.dist;
prop.the[1]=(zb-za)/prop.dist;
prop.the[0]=prop.the[1]-q;
prop.the[1]=-prop.the[1]-q;
prop.dl[0]=prop.dist;
prop.dl[1]=prop.dist;

if (np>=2)
{
    sa=0.0;
    sb=prop.dist;
    wq=true;
}

```

```

for (int i=1; i<np; i++)
{
    sa+=xi;
    sb-=xi;
    q=pfl[i+2]-(qc*sa+prop.the[0])*sa-za;

    if (q>0.0)
    {
        prop.the[0]+=q/sa;
        prop.dl[0]=sa;
        wq=false;
    }

    if (!wq)
    {
        q=pfl[i+2]-(qc*sb+prop.the[1])*sb-zb;

        if (q>0.0)
        {
            prop.the[1]+=q/sb;
            prop.dl[1]=sb;
        }
    }
}
}

```

```

void hzns2(double pfl[], prop_type &prop, propa_type &propa)
{
    bool wq;
    int np, rp, i, j;
    double xi, za, zb, qc, q, sb, sa, dr, dshh;

    np=(int)pfl[0];
    xi=pfl[1];
    za=pfl[2]+prop.hg[0];
    zb=pfl[np+2]+prop.hg[1];

```

```

prop.tiw=xi;
prop.ght=za;
prop.ghr=zb;
qc=0.5*prop.gme;
q=qc*prop.dist;
prop.the[1]=atan((zb-za)/prop.dist);
prop.the[0]=(prop.the[1])-q;
prop.the[1]=-prop.the[1]-q;
prop.dl[0]=prop.dist;
prop.dl[1]=prop.dist;
prop.hht=0.0;
prop.hhr=0.0;
prop.los=1;

```

```

if(np>=2)
{
sa=0.0;
sb=prop.dist;
wq=true;

for(j=1; j<np; j++)
{
sa+=xi;
q=pfl[j+2]-(qc*sa+prop.the[0])*sa-za;

```

```

if(q>0.0)
{
prop.los=0;
prop.the[0]+=q/sa;
prop.dl[0]=sa;
prop.the[0]=mymin(prop.the[0],1.569);
prop.hht=pfl[j+2];
wq=false;
}
}
```

```

if(!wq)
{
for(i=1; i<np; i++)

```

```

{
sb-=xi;
q=pfl[np+2-i]-(qc*(prop.dist-sb)+prop.the[1])*(prop.dist-sb)-zb;
if(q>0.0)
{
prop.the[1]+=q/(prop.dist-sb);
prop.the[1]=mymin(prop.the[1],1.57);
prop.the[1]=mymax(prop.the[1],-1.568);
prop.hhr=pfl[np+2-i];
prop.dl[1]=mymax(0.0,prop.dist-sb);
}
}

prop.the[0]=atan((prop.hht-za)/prop.dl[0])-0.5*prop.gme*prop.dl[0];
prop.the[1]=atan((prop.hhr-zb)/prop.dl[1])-0.5*prop.gme*prop.dl[1];
}

}

if((prop.dl[1])<(prop.dist))
{
dshh=prop.dist-prop.dl[0]-prop.dl[1];

if(int(dshh)==0) /* one obstacle */
{
dr=prop.dl[1]/(1+zb/prop.hht);
}
else /* two obstacles */
{
dr=prop.dl[1]/(1+zb/prop.hhr);
}
}

else /* line of sight */
{
dr=(prop.dist)/(1+zb/za);
}
rp=2+(int)(floor(0.5+dr/xi));
prop.rpl=rp;
prop.rph=pfl[rp];
}

```

```

void z1sq1 (double z[], const double &x1, const double &x2, double& z0, double& zn)
{
/* Used only with ITM 1.2.2 */
double xn, xa, xb, x, a, b;
int n, ja, jb;

xn=z[0];
xa=int(FORTRAN_DIM(x1/z[1],0.0));
xb=xn-int(FORTRAN_DIM(xn,x2/z[1]));

if (xb<=xa)
{
xa=FORTRAN_DIM(xa,1.0);
xb=xn-FORTRAN_DIM(xn,xb+1.0);
}

ja=(int)xa;
jb=(int)xb;
n=jb-ja;
xa=xb-xa;
x=-0.5*xa;
xb+=x;
a=0.5*(z[ja+2]+z[jb+2]);
b=0.5*(z[ja+2]-z[jb+2])*x;

for (int i=2; i<=n; ++i)
{
++ja;
x+=1.0;
a+=z[ja+2];
b+=z[ja+2]*x;
}

a/=xa;
b=b*12.0/((xa*xa+2.0)*xa);
z0=a-b*xb;
zn=a+b*(xn-xb);
}

```

```

void z1sq2(double z[], const double &x1, const double &x2, double& z0, double& zn)
{
/* corrected for use with ITWOM */
double xn, xa, xb, x, a, b, bn;
int n, ja, jb;

xn=z[0];
xa=int(FORTRAN_DIM(x1/z[1],0.0));
xb=xn-int(FORTRAN_DIM(xn,x2/z[1]));

if (xb<=xa)
{
xa=FORTRAN_DIM(xa,1.0);
xb=xn-FORTRAN_DIM(xn,xb+1.0);
}

ja=(int)xa;
jb=(int)xb;
xa=(2*int((xb-xa)/2))-1;
x=-0.5*(xa+1);
xb+=x;
ja=jb-1-(int)xa;
n=jb-ja;
a=(z[ja+2]+z[jb+2]);
b=(z[ja+2]-z[jb+2])*x;
bn=2*(x*x);

for (int i=2; i<=n; ++i)
{
++ja;
x+=1.0;
bn+=(x*x);
a+=z[ja+2];
b+=z[ja+2]*x;
}

a/=(xa+2);
b=b/bn;

```

```

z0=a-(b*xb);
zn=a+(b*(xn-xb));
}

double qtile (const int &nn, double a[], const int &ir)
{
    double q=0.0, r; /* q initialization -- KD2BD */
    int m, n, i, j, j1=0, i0=0, k; /* more initializations -- KD2BD */
    bool done=false;
    bool goto10=true;

    m=0;
    n=nn;
    k=mymin(mymax(0,ir),n);

    while (!done)
    {
        if (goto10)
        {
            q=a[k];
            i0=m;
            j1=n;
        }

        i=i0;

        while (i<=n && a[i]>=q)
            i++;

        if (i>n)
            i=n;

        j=j1;

        while (j>=m && a[j]<=q)
            j--;

        if (j<m)
            j=m;
    }
}

```

```

if (i<j)
{
    r=a[i];
    a[i]=a[j];
    a[j]=r;
    i0=i+1;
    j1=j-1;
    goto10=false;
}

else if (i<k)
{
    a[k]=a[i];
    a[i]=q;
    m=i+1;
    goto10=true;
}

else if (j>k)
{
    a[k]=a[j];
    a[j]=q;
    n=j-1;
    goto10=true;
}

else
done=true;
}

return q;
}

double qerf(const double &z)
{
double b1=0.319381530, b2=-0.356563782, b3=1.781477937;
double b4=-1.821255987, b5=1.330274429;
double rp=4.317008, rrt2pi=0.398942280;

```

```

double t, x, qerfv;

x=z;
t=fabs(x);

if (t>=10.0)
qerfv=0.0;
else
{
t=rp/(t+rp);
qerfv=exp(-0.5*x*x)*rrt2pi*(((b5*t+b4)*t+b3)*t+b2)*t+b1)*t;
}

if (x<0.0)
qerfv=1.0-qerfv;

return qerfv;
}

```

```

double d1thx(double pfl[], const double &x1, const double &x2)
{
int np, ka, kb, n, k, j;
double d1thxv, sn, xa, xb;
double *s;

np=(int)pfl[0];
xa=x1/pfl[1];
xb=x2/pfl[1];
d1thxv=0.0;

if (xb-xa<2.0) // exit out
return d1thxv;

ka=(int)(0.1*(xb-xa+8.0));
ka=mymin(mymax(4,ka),25);
n=10*ka-5;
kb=n-ka+1;
sn=n-1;

```

```

assert((s=new double[n+2])!=0);
s[0]=sn;
s[1]=1.0;
xb=(xb-xa)/sn;
k=(int)(xa+1.0);
xa-=(double)k;

for (j=0; j<n; j++)
{
    while (xa>0.0 && k<np)
    {
        xa-=1.0;
        ++k;
    }

    s[j+2]=pfl[k+2]+(pfl[k+2]-pfl[k+1])*xa;
    xa=xa+xb;
}

z1sq1(s,0.0,sn,xa,xb);
xb=(xb-xa)/sn;

for (j=0; j<n; j++)
{
    s[j+2]-=xa;
    xa=xa+xb;
}

d1thxv=qtile(n-1,s+2,ka-1)-qtile(n-1,s+2,kb-1);
d1thxv/=1.0-0.8*exp(-(x2-x1)/50.0e3);
delete[] s;

return d1thxv;
}

double d1thx2(double pfl[], const double &x1, const double &x2, propa_type &propa)
{
    int np, ka, kb, n, k, kmx, j;
}

```

```

double d1thx2v, sn, xa, xb, xc;
double *s;

np=(int)pfl[0];
xa=x1/pfl[1];
xb=x2/pfl[1];
d1thx2v=0.0;

if (xb-xa<2.0) // exit out
return d1thx2v;

ka=(int)(0.1*(xb-xa+8.0));
kmx=mymax(25,(int)(83350/(pfl[1])));
ka=mymin(mymax(4,ka),kmx);
n=10*ka-5;
kb=n-ka+1;
sn=n-1;
assert((s=new double[n+2])!=0);
s[0]=sn;
s[1]=1.0;
xb=(xb-xa)/sn;
k=(int(xa+1.0));
xc=xa-(double(k));

for (j=0; j<n; j++)
{
while (xc>0.0 && k<np)
{
xc-=1.0;
++k;
}

s[j+2]=pfl[k+2]+(pfl[k+2]-pfl[k+1])*xc;
xc=xc+xb;
}

z1sq2(s,0.0,sn,xa,xb);
xb=(xb-xa)/sn;

```

```

for (j=0; j<n; j++)
{
    s[j+2]-=xa;
    xa=xa+xb;
}

d1thx2v=qtile(n-1,s+2,ka-1)-qtile(n-1,s+2,kb-1);
d1thx2v/=1.0-0.8*exp(-(x2-x1)/50.0e3);
delete[] s;
return d1thx2v;
}

void qlrpfl(double pfl[], int klimx, int mdvarx, prop_type &prop, propa_type &propa, propv_type
&propv)
{
    int np, j;
    double xl[2], q, za, zb, temp;

    prop.dist=pfl[0]*pfl[1];
    np=(int)pfl[0];
    hzns(pfl,prop);

    for (j=0; j<2; j++)
        xl[j]=mymin(15.0*prop.hg[j],0.1*prop.dl[j]);

    xl[1]=prop.dist-xl[1];
    prop.dh=d1thx(pfl,xl[0],xl[1]);

    if (prop.dl[0]+prop.dl[1]>1.5*prop.dist)
    {
        z1sq1(pfl,xl[0],xl[1],za,zb);
        prop.he[0]=prop.hg[0]+FORTRAN_DIM(pfl[2],za);
        prop.he[1]=prop.hg[1]+FORTRAN_DIM(pfl[np+2],zb);

        for (j=0; j<2; j++)
            prop.dl[j]=sqrt(2.0*prop.he[j]/prop.gme)*exp(-0.07*sqrt(prop.dh/mymax(prop.he[j],5.0)));

        q=prop.dl[0]+prop.dl[1];
    }
}

```

```

if (q<=prop.dist) /* if there is a rounded horizon, or two obstructions, in the path */
{
/* q=pow(prop.dist/q,2.0); */
temp=prop.dist/q;
q=temp*temp;

for (j=0; j<2; j++)
{
prop.he[j]*=q; /* tx effective height set to be path dist/distance between obstacles */
prop.dl[j]=sqrt(2.0*prop.he[j]/prop.gme)*exp(-0.07*sqrt(prop.dh/mymax(prop.he[j],5.0)));
}
}

for (j=0; j<2; j++) /* original empirical adjustment? uses delta-h to adjust grazing angles */
{
q=sqrt(2.0*prop.he[j]/prop.gme);
prop.the[j]=(0.65*prop.dh*(q/prop.dl[j]-1.0)-2.0*prop.he[j])/q;
}
}

else
{
z1sq1(pfl,xl[0],0.9*prop.dl[0],za,q);
z1sq1(pfl,prop.dist-0.9*prop.dl[1],xl[1],q,zb);
prop.he[0]=prop.hg[0]+FORTRAN_DIM(pfl[2],za);
prop.he[1]=prop.hg[1]+FORTRAN_DIM(pfl[np+2],zb);
}

prop.mdp=-1;
propv.lvar=mymax(propv.lvar,3);

if (mdvarx>=0)
{
propv.mdvar=mdvarx;
propv.lvar=mymax(propv.lvar,4);
}

if (klimx>0)

```

```

{
propv.klim=klimx;
propv.lvar=5;
}

lrprop(0.0,prop,propa);
}

void qlrpfl2(double pfl[], int klimx, int mdvarx, prop_type &prop, propa_type &propa, propv_type
&propv)
{
int np, j;
double xl[2], dlb, q, za, zb, temp, rad, rae1, rae2;

prop.dist=pfl[0]*pfl[1];
np=(int)pfl[0];
hzns2(pfl,prop, propa);
dlb=prop.dl[0]+prop.dl[1];
prop.rch[0]=prop.hg[0]+pfl[2];
prop.rch[1]=prop.hg[1]+pfl[np+2];

for (j=0; j<2; j++)
xl[j]=mymin(15.0*prop.hg[j],0.1*prop.dl[j]);

xl[1]=prop.dist-xl[1];
prop.dh=d1thx2(pfl,xl[0],xl[1],propa);

if ((np<1) || (pfl[1]>150.0))
{
/* for TRANSHORIZON; diffraction over a mutual horizon, or for one or more obstructions */
if (dlb<1.5*prop.dist)
{
z1sq2(pfl,xl[0],0.9*prop.dl[0],za,q);
z1sq2(pfl,prop.dist-0.9*prop.dl[1],xl[1],q,zb);
prop.he[0]=prop.hg[0]+FORTRAN_DIM(pfl[2],za);
prop.he[1]=prop.hg[1]+FORTRAN_DIM(pfl[np+2],zb);
}
}

/* for a Line-of-Sight path */

```

```

else
{
z1sq2(pfl,xl[0],xl[1],za,zb);
prop.he[0]=prop.hg[0]+FORTRAN_DIM(pfl[2],za);
prop.he[1]=prop.hg[1]+FORTRAN_DIM(pfl[np+2],zb);

for (j=0; j<2; j++)
prop.dl[j]=sqrt(2.0*prop.he[j]/prop.gme)*exp(-0.07*sqrt(prop.dh/mymax(prop.he[j],5.0)));

/* for one or more obstructions only NOTE buried as in ITM FORTRAN and DLL, not functional */
if ((prop.dl[0]+prop.dl[1])<=prop.dist)
{
/* q=pow(prop.dist/(dl[0]+dl[1]),2.0); */
temp=prop.dist/(prop.dl[0]+prop.dl[1]);
q=temp*temp;
}

for (j=0; j<2; j++)
{
prop.he[j]*=q;
prop.dl[j]=sqrt(2.0*prop.he[j]/prop.gme)*exp(-0.07*sqrt(prop.dh/mymax(prop.he[j],5.0)));
}

/* this sets (or resets) prop.the, and is not in The Guide FORTRAN QLRPFL */
for (j=0; j<2; j++)
{
q=sqrt(2.0*prop.he[j]/prop.gme);
prop.the[j]=(0.65*prop.dh*(q/prop.dl[j]-1.0)-2.0*prop.he[j])/q;
}
}

else /* for ITWOM ,computes he for tx, rcvr, and the receiver approach angles for use in saalos */
{
prop.he[0]=prop.hg[0]+(pfl[2]);
prop.he[1]=prop.hg[1]+(pfl[np+2]);

rad=(prop.dist-500.0);
}

```

```

if (prop.dist>550.0)
{
  z1sq2(pfl,rad,prop.dist,rae1,rae2);
}
else
{
  rae1=0.0;
  rae2=0.0;
}

prop.thera=atan(abs(rae2-rae1)/prop.dist);

if (rae2<rae1)
{
  prop.thera=-prop.thera;
}

prop.thenr=atan(mymax(0.0,(pfl[np+2]-pfl[np+1])/pfl[1]));

}

prop.mdp=-1;
propv.lvar=mymax(propv.lvar,3);

if (mdvarx>=0)
{
  propv.mdvar=mdvarx;
  propv.lvar=mymax(propv.lvar,4);
}

if (klimx>0)
{
  propv.klim=klimx;
  propv.lvar=5;
}

lrprop2(0.0,prop,propa);
}

```

```
double deg2rad(double d)
{
    return d*3.1415926535897/180.0;
}

//*****
/* Point-To-Point Mode Calculations
//*****
```

```
void point_to_point(double elev[], double tht_m, double rht_m, double eps_dielect, double
sgm_conductivity, double eno_ns_surfref, double frq_mhz, int radio_climate, int pol, double conf,
double rel, double &dbloss, char *strmode, int &errnum)
```

```
*****
```

pol:

0-Horizontal, 1-Vertical

radio_climate:

1-Equatorial, 2-Continental Subtropical,
3-Maritime Tropical, 4-Desert, 5-Continental Temperate,
6-Maritime Temperate, Over Land, 7-Maritime Temperate,
Over Sea

conf, rel: .01 to .99

elev[]: [num points - 1], [delta dist(meters)],
[height(meters) point 1], ..., [height(meters) point n]

errnum: 0- No Error.

1- Warning: Some parameters are nearly out of range.

Results should be used with caution.

2- Note: Default parameters have been substituted for
impossible ones.

3- Warning: A combination of parameters is out of range.
Results are probably invalid.

Other- Warning: Some parameters are out of range.
Results are probably invalid.

```

*****
/
{
prop_type prop;
propv_type propv;
propa_type propa;
double zsys=0;
double zc, zr;
double eno, enso, q;
long ja, jb, i, np;
double dkm, xkm;
double fs;

prop.hg[0]=tht_m;
prop.hg[1]=rht_m;
propv.klim=radio_climate;
prop.kwx=0;
propv.lvar=5;
prop.mdp=-1;
zc=qerfi(conf);
zr=qerfi(rel);
np=(long)elev[0];
dkm=(elev[1]*elev[0])/1000.0;
xkm=elev[1]/1000.0;
eno=eno_ns_surfref;
enso=0.0;
q=enso;

if (q<=0.0)
{
ja=(long)(3.0+0.1*elev[0]); /* added (long) to correct */
jb=np-ja+6;

for (i=ja-1; i<jb; ++i)
zsys+=elev[i];

zsys/=(jb-ja+1);
q=eno;
}

```

```

propv.mdvar=12;
qlrps(frq_mhz,zsys,q,pol,eps_dielect,sgm_conductivity,prop);
qlrpfl(elev,propv.klim,propv.mdvar,prop,propa,propv);
fs=32.45+20.0*log10(frq_mhz)+20.0*log10(prop.dist/1000.0);
q=prop.dist-propa.dla;

if (int(q)<0.0)
strcpy(strmode,"Line-Of-Sight Mode");
else
{
if (int(q)==0.0)
strcpy(strmode,"Single Horizon");

else if (int(q)>0.0)
strcpy(strmode,"Double Horizon");

if (prop.dist<=propa.dlsa || prop.dist <= propa.dx)
strcat(strmode,", Diffraction Dominant");

else if (prop.dist>propa.dx)
strcat(strmode, ", Troposcatter Dominant");
}

dbloss=avar(zr,0.0,zc,prop,propv)+fs;
errnum=prop.kwx;
}

void point_to_point_two(double elev[], double tht_m, double rht_m, double eps_dielect, double
sgm_conductivity, double eno_ns_surref, double enc_ncc_clcref, double clutter_height, double
clutter_density, double delta_h_diff, double frq_mhz, int radio_climate, int pol, int mode_var, double
conf, double rel, double loc, double &dbloss, char *strmode, int &errnum, double &dlsa, double
&test0, double &test1, double &test2, double &test3, double &test4)

*****
pol:

```

0-Horizontal, 1-Vertical, 2-Circular

radio_climate:

1-Equatorial, 2-Continental Subtropical,
3-Maritime Tropical, 4-Desert, 5-Continental Temperate,
6-Maritime Temperate, Over Land, 7-Maritime Temperate,
Over Sea

conf, rel: .01 to .99

elev[]: [num points - 1], [delta dist(meters)],
[height(meters) point 1], ..., [height(meters) point n]

clutter_height 25.2 meters for compatibility with ITU-R P.1546-2.

clutter_density 1.0 for compatibility with ITU-R P.1546-2.

delta_h_diff optional delta h for beyond line of sight. 90 m. average.
setting to 0.0 will default to use of original internal
use of delta-h for beyond line-of-sight range.

mode_var set to 12; or to 1 for FCC ILLR; see documentation

enc_ncc_clcref clutter refractivity; 1000 N-units to match ITU-R P.1546-2

eno=eno_ns_surfref atmospheric refractivity at sea level; 301 N-units nominal
(ranges from 250 for dry, hot day to 450 on hot, humid day]
(stabilizes near 301 in cold, clear weather)

errnum: 0- No Error.

1- Warning: Some parameters are nearly out of range.

Results should be used with caution.

2- Note: Default parameters have been substituted for
impossible ones.

3- Warning: A combination of parameters is out of range.

Results are probably invalid.

Other- Warning: Some parameters are out of range.

Results are probably invalid.

```
*****
{
prop_type prop;
propv_type propv;
propa_type propa;
double zsys=0;
double zc, zr;
double eno, enso, q;
long ja, jb, i, np;
double dkm, xkm;
double tpd, fs;
prop.hg[0]=tht_m;
prop.hg[1]=rht_m;
propv.klim=radio_climate;
prop.encc=enc_ncc_clcref;
prop.cch=clutter_height;
prop.cd=clutter_density;
prop.dhd=delta_h_diff;
prop.kwx=0;
propv.lvar=5;
prop.mdp=-1;
prop.ptx=pol;
prop.thera=0.0;
prop.thenr=0.0;
zc=qerfi(conf);
zr=qerfi(rel);
np=(long)elev[0];
dkm=(elev[1]*elev[0])/1000.0;
xkm=elev[1]/1000.0;
eno=eno_ns_surfref;
enso=0.0;
q=enso;
```

/ PRESET VALUES for Basic Version w/o additional inputs active */*

```
prop.encc = 1000.00; /* double enc_ncc_clcref */
prop.cch = 22.5; /* double clutter_height */
prop.cd = 1.00; /* double clutter_density */
mode_var =1; /* int mode_var set for FCC ILLR */
```

```

loc =0.50; /* double loc */

if (q<=0.0)
{
ja=(long)(3.0+0.1*elev[0]);
jb=np-ja+6;

for (i=ja-1; i<jb; ++i)
zsys+=elev[i];

zsys/=(jb-ja+1);
q=eno;
}

propv.mdvar=mode_var;
qlrps(frq_mhz,zsys,q,pol,eps_dielect,sgm_conductivity,prop);
qlrpfl2(elev,propv.klim,propv.mdvar,prop,propa,propv);
tpd=sqrt((prop.he[0]-prop.he[1])*(prop.he[0]-prop.he[1])+(prop.dist)*(prop.dist));
fs=32.45+20.0*log10(frq_mhz)+20.0*log10(tpd/1000.0);
/*propa.test4=fs; */

q=prop.dist-propa.dla;

if (int(q)<0.0)
strcpy(strmode,"L-o-S");
else
{
if (int(q)==0.0)
strcpy(strmode,"1_Hrzn");

else if (int(q)>0.0)
strcpy(strmode,"2_Hrzn");

if (prop.dist<=propa.dlsa || prop.dist<=propa.dx)

if(int(prop.dl[1]==0.0)
strcat(strmode,"_Peak");

else

```

```

strcat(strmode,"_Diff");

else if (prop.dist>propa.dx)
strcat(strmode, "_Tropo");
}

dbloss=avar(zr,0.0,zc,prop,propv)+fs;
errnum=prop.kwx;
dlsa=propa.dlsa;
test0=propa.test0;
test1=propa.test1;
test2=propa.test2;
test3=propa.test3;
test4=propa.test4;
}

```

```

void point_to_pointMDH_two (double elev[], double tht_m, double rht_m,
    double eps_dielect, double sgm_conductivity, double eno_ns_surfref,
    double enc_ncc_clcref, double clutter_height, double clutter_density,
    double delta_h_diff, double frq_mhz, int radio_climate, int pol, int mode_var,
    double timepct, double locpct, double confpct,
    double &dbloss, int &propmode, double &deltaH, int &errnum)

```

pol: 0-Horizontal, 1-Vertical

radio_climate: 1-Equatorial, 2-Continental Subtropical, 3-Maritime Tropical,

4-Desert, 5-Continental Temperate, 6-Maritime Temperate, Over Land,

7-Maritime Temperate, Over Sea

timepct, locpct, confpct: .01 to .99

elev[]: [num points - 1], [delta dist(meters)], [height(meters) point 1], ..., [height(meters) point n]

propmode: Value Mode

- 1 mode is undefined

- 0 Line of Sight

- 5 Single Horizon, Diffraction

- 6 Single Horizon, Troposcatter

- 9 Double Horizon, Diffraction

- 10 Double Horizon, Troposcatter

errnum: 0- No Error.

- 1- Warning: Some parameters are nearly out of range.
Results should be used with caution.
- 2- Note: Default parameters have been substituted for impossible ones.
- 3- Warning: A combination of parameters is out of range.
Results are probably invalid.
- Other- Warning: Some parameters are out of range.
Results are probably invalid.

```
******/  
{
```

```
prop_type prop;  
propv_type propv;  
propa_type propa;  
double zsys=0;  
double ztime, zloc, zconf;  
double eno, enso, q;  
long ja, jb, i, np;  
double dkm, xkm;  
double fs;  
  
propmode = -1; // mode is undefined  
prop.hg[0] = tht_m;  
prop.hg[1] = rht_m;  
propv.klim = radio_climate;  
prop.encc=enc_ncc_clcref;  
prop.cch=clutter_height;  
prop.cd=clutter_density;  
prop.dhd=delta_h_diff;  
prop.kwx = 0;  
propv.lvar = 5;  
prop.mdp = -1;  
prop.ptx=pol;  
prop.thera=0.0;  
prop.thenr=0.0;  
ztime = qerfi(timepct);  
zloc = qerfi(locpct);  
zconf = qerfi(confpct);  
np = (long)elev[0];  
dkm = (elev[1] * elev[0]) / 1000.0;
```

```

xkm = elev[1] / 1000.0;
eno = eno_ns_surfref;
enso = 0.0;
q = enso;

/* PRESET VALUES for Basic Version w/o additional inputs active */

prop.encc = 1000.00; /* double enc_ncc_clcref */
prop.cch = 22.5; /* double clutter_height */
prop.cd = 1.00; /* double clutter_density */
mode_var = 1; /* int mode_var set for FCC ILLR */

if(q<=0.0)
{
    ja =(long) (3.0 + 0.1 * elev[0]); /* to match addition of (long) */
    jb = np - ja + 6;
    for(i=ja-1;i<jb;++)
    {
        zsys+=elev[i];
        zsys/=(jb-ja+1);
    }
    q=eno;
}
propv.mdvar=12;
qlrps(frq_mhz,zsys,q,pol,eps_dielect,sgm_conductivity,prop);
qlrpfl2(elev,propv.klim,propv.mdvar,prop,propa,propv);
fs=32.45+20.0*log10(frq_mhz)+20.0*log10(prop.dist/1000.0);

/*propa.test4=fs;*/
deltaH = prop.dh;
q = prop.dist - propa.dla;
if(int(q)<0.0)
    propmode = 0; // L-of-S
else
    { if(int(q)==0.0)
        propmode = 4; // 1-Hrzn
        else if(int(q)>0.0)
            propmode = 8; // 2-Hrzn
        if(prop.dist<=propa.dlsa || prop.dist<=propa.dx)
            propmode += 1; // Diff
        else if(prop.dist>propa.dx)

```

```

    propmode += 2; // Tropo
}
dbloss = avar(ztime, zloc, zconf, prop, propv) + fs; //avar(time,location,confidence)
errnum = prop.kwx;
}

void point_to_pointDH (double elev[], double tht_m, double rht_m,
    double eps_dielect, double sgm_conductivity, double eno_ns_surfref,
    double enc_ncc_clcref, double clutter_height, double clutter_density, double delta_h_diff,
    double frq_mhz, int radio_climate, int pol, double conf, double rel, double loc,
    double &dbloss, double &deltaH, int &errnum)

/********************************************

pol: 0-Horizontal, 1-Vertical
radio_climate: 1-Equatorial, 2-Continental Subtropical, 3-Maritime Tropical,
    4-Desert, 5-Continental Temperate, 6-Maritime Temperate, Over Land,
    7-Maritime Temperate, Over Sea
conf, rel: .01 to .99
elev[]: [num points - 1], [delta dist(meters)], [height(meters) point 1], ..., [height(meters) point n]
errnum: 0- No Error.

    1- Warning: Some parameters are nearly out of range.
        Results should be used with caution.
    2- Note: Default parameters have been substituted for impossible ones.
    3- Warning: A combination of parameters is out of range.
        Results are probably invalid.

Other- Warning: Some parameters are out of range.
    Results are probably invalid.

********************************************/
{

char strmode[100];
prop_type prop;
propv_type propv;
propa_type propa;
double zsys=0;
double zc, zr;
double eno, enso, q;
long ja, jb, i, np;
double dkm, xkm;
double fs;

```

```

prop.hg[0]=tht_m;
prop.hg[1]=rht_m;
propv.klim = radio_climate;
prop.encc=enc_ncc_clcref;
prop.cch=clutter_height;
prop.cd=clutter_density;
prop.dhd=delta_h_diff;
prop.kwx = 0;
propv.lvar = 5;
prop.mdp = -1;
prop.ptx=pol;
prop.thera=0.0;
prop.thenr=0.0;
zc = qerfi(conf);
zr = qerfi(rel);
np = (long)elev[0];
dkm = (elev[1] * elev[0]) / 1000.0;
xkm = elev[1] / 1000.0;
eno = eno_ns_surfreq;
enso = 0.0;
q = enso;

```

/* PRESET VALUES for Basic Version w/o additional inputs active */

```

prop.encc = 1000.00; /* double enc_ncc_clcref */
prop.cch = 22.5;    /* double clutter_height */
prop.cd = 1.00;      /* double clutter_density */

if(q<=0.0)
{
    ja = (long) (3.0 + 0.1 * elev[0]); /* to match KD2BD addition of (long) */
    jb = np - ja + 6;
    for(i=ja-1; i<jb; ++i)
        zsys+=elev[i];
    zsys/=(jb-ja+1);
    q=eno;
}
propv.mdvar=12;

```

```

qlrps(frq_mhz,zsys,q,pol,eps_dielect,sgm_conductivity,prop);
qlrpfl2(elev,propv.klim,propv.mdvar,prop,propa,propv);
fs=32.45+20.0*log10(frq_mhz)+20.0*log10(prop.dist/1000.0);
/*propa.test4=fs; */
deltaH = prop.dh;
q = prop.dist - propa.dla;
if(int(q)<0.0)
    strcpy(strmode,"Line-Of-Sight Mode");
else
{ if(int(q)==0.0)
    strcpy(strmode,"Single Horizon");
else if(int(q)>0.0)
    strcpy(strmode,"Double Horizon");
if(prop.dist<=propa.dlsa || prop.dist<=propa.dx)
    strcat(strmode,", Diffraction Dominant");
else if(prop.dist>propa.dx)
    strcat(strmode, ", Troposcatter Dominant");
}
dbloss = avar(zr,0.0,zc,prop,propv)+fs; //avar(time,location,confidence)
errnum = prop.kwx;
}

//*****
//** Area Mode Calculations
//*****

```

```

void area(long ModVar, double deltaH, double tht_m, double rht_m, double dist_km, int TSiteCriteria,
int RSiteCriteria, double eps_dielect, double sgm_conductivity, double eno_ns_surfref, double
enc_ncc_clcref, double clutter_height, double clutter_density, double delta_h_diff, double frq_mhz, int
radio_climate, int pol, int mode_var,
double pctTime, double pctLoc, double pctConf, double &dbloss, char *strmode, int &errnum)
{
// pol: 0-Horizontal, 1-Vertical
// TSiteCriteria, RSiteCriteria:
// 0 - random, 1 - careful, 2 - very careful

// radio_climate: 1-Equatorial, 2-Continental Subtropical, 3-Maritime Tropical,
// 4-Desert, 5-Continental Temperate, 6-Maritime Temperate, Over Land,

```

```

//      7-Maritime Temperate, Over Sea
// ModVar: 0 - Single: pctConf is "Time/Situation/Location", pctTime, pctLoc not used
//      1 - Individual: pctTime is "Situation/Location", pctConf is "Confidence", pctLoc not used
//      2 - Mobile: pctTime is "Time/Locations (Reliability)", pctConf is "Confidence", pctLoc not
used
//      3 - Broadcast: pctTime is "Time", pctLoc is "Location", pctConf is "Confidence"
// pctTime, pctLoc, pctConf: .01 to .99
// errnum: 0- No Error.
//      1- Warning: Some parameters are nearly out of range.
//            Results should be used with caution.
//      2- Note: Default parameters have been substituted for impossible ones.
//      3- Warning: A combination of parameters is out of range.
//            Results are probably invalid.
// Other- Warning: Some parameters are out of range.
//            Results are probably invalid.
// NOTE: strmode is not used at this time.

```

```

prop_type prop;
propv_type propv;
propa_type propa;
double zt, zl, zc, xl;
double fs;
long ivar;
double eps, eno, sgm;
long ipol;
int kst[2];

```

```

kst[0]=(int)TSiteCriteria;
kst[1]=(int)RSiteCriteria;
zt=qerfi(pctTime/100.0);
zl=qerfi(pctLoc/100.0);
zc=qerfi(pctConf/100.0);
eps=eps_dielect;
sgm=sgm_conductivity;
eno=eno_ns_surfref;
prop.dh=deltaH;
prop.hg[0]=tht_m;
prop.hg[1]=rht_m;
propv.klim=(long)radio_climate;

```

```

prop.encc=enc_ncc_clcref;
prop.cch=clutter_height;
prop.cd=clutter_density;
prop.dhd=delta_h_diff;
prop.ens=eno;
prop.kwx=0;
ivar=(long)ModVar;
ipol=(long)pol;
qlrps(frq_mhz, 0.0, eno, ipol, eps, sgm, prop);
qlra(kst, propv.klim, ivar, prop, propv);

if (propv.lvar<1)
propv.lvar=1;

lrprop2(dist_km*1000.0, prop, propa);
fs=32.45+20.0*log10(frq_mhz)+20.0*log10(prop.dist/1000.0);
/* propa.test4=fs; */
xlb=fs+avar(zt, zl, xc, prop, propv);
dbloss=xlb;
if (prop.kwx==0)
errnum=0;
else
errnum=prop.kwx;
}

```

```

double ITMReadBLoss(long ModVar, double deltaH, double tht_m, double rht_m,
double dist_km, int TSiteCriteria, int RSiteCriteria,
double eps_dielect, double sgm_conductivity, double eno_ns_surfref,
double enc_ncc_clcref,double clutter_height,double clutter_density,
double delta_h_diff, double frq_mhz, int radio_climate, int pol,
int mode_var, double pctTime, double pctLoc, double pctConf)
{
char strmode[200];
int errnum;
double dbloss;
area(ModVar,deltaH,tht_m,rht_m,dist_km,TSiteCriteria,RSiteCriteria,
eps_dielect,sgm_conductivity,eno_ns_surfref,
enc_ncc_clcref,clutter_height,clutter_density,

```

```
    delta_h_diff,frq_mhz,radio_climate,pol,mode_var,pctTime,  
    pctLoc,pctConf,dbloss,strmode,errnum);  
    return dbloss;  
}
```

```
double ITWOMVersion()  
{  
    return 2.0;  
}
```